File: `bisection.8th`

Find a zero of the given function $f(x)$ in the interval (a, b). For best performance, we should have $f(a)f(b) < 0$, i.e. the values should have opposite signs.

Set `num-debug` to true, if you want to see the iterations.

Set `lines` to something like 5 or 10 if you have to much output.

If you need more accuracy than the machine accuracy of 15 secure places, you can use something like `30 big-floats`.

You should also consult the documentation of the numerics library.

```
needs tools
needs numerics
with: nm
with: n
```

We store $f(a)$ to not have to evaluate it twice.

```
var f(a)
```

This is a test function: $f(x) = x^3 + 4x^2 - 10$. We evaluate it using Horner's method: $(x + 4)x^2 - 10$.

```
: f-test \ n -- n
   dup 4 + swap sqr * 10 - ;
```

Calculate the midpoint between a and b using $p = a + \frac{b-a}{2}$, which is safer against overflow than $\frac{a+b}{2}$.

```
: midpoint \ a b -- a+(b-a)/2
   tuck - 2 / 4 add-item + ;
```

Exit criterion. Returns true iff $f(p) < \epsilon$ or $|b - p| < \epsilon$. criterion n b p f(p) – n b p f(p) ?

```
: bis-criterion \ a b n -- a b n
   row @ 3 a:@ 0~ swap 4 a:@ 0~ nip or ;
```

Takes an interval (a, b) and returns a new interval, which is either (a, p) or (p, b) depending on which the zero lies in. next-interval a b n – a p — p b

```
: bis-algorithm \ a b n -- a p n | p b n
   >r
   1 add-item
   swap 0 add-item \ b a
   2dup midpoint 2 add-item \ b a p
   dup f 3 add-item \ b a p f(p)
   f(a) @ over \ b a p f(p) f(a) f(p)
   * 0 > if \ b a p f(p)
      f(a) ! nip swap
   else drop rot drop then
   r> ;
```

Given an initial interval, that includes at least one zero, `bisection` returns an interval with a zero in it.

```
: bisection \ a b -- a b
   over f f(a) !
   ["a","b","p","f(p)","|b-a|/2"] main ;
```

Show the steps

```
true nm:num-debug !

' f-test w:is nm:f
' bis-criterion w:is criterion?
' bis-algorithm w:is algorithm
F1. 2. bisection f. space f. cr
;with ;with bye
```